ADTool: Security Analysis with Attack–Defense Trees
Extended Version*

Barbara Kordy, Piotr Kordy, Sjouke Mauw, Patrick Schweitzer
{barbara.kordy, piotr.kordy, sjouke.mauw, patrick.schweitzer}@uni.lu

University of Luxembourg, SnT
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg

Abstract. The ADTool is free, open source software assisting graphical modeling and quantitative analysis of security, using attack–defense trees. The main features of the ADTool are easy creation, efficient editing, and automated bottom-up evaluation of security-relevant measures. The tool also supports the usage of attack trees, protection trees and defense trees, which are all particular instances of attack–defense trees.

1 Introduction

Graphical security models provide an intuitive but systematic methodology to visualize possible attacks and countermeasures and to enable the computation of various security related parameters. Thus, they have been widely accepted by the industrial sector, as a means to support and facilitate threat analysis and risk management processes. Graphical models have been used for security analysis of SCADA systems [8,32], of voting systems [23,7], of vehicular communication systems [12,2], of Internet related attacks [33,24], and of socio-technical attacks [4,11,30]. This paper presents the ADTool software [21] which supports quantitative and qualitative security assessment using a graphical security modeling technique called attack–defense trees.

1.1 Background

Attack–defense trees (ADTrees) [17] extend and improve the well-known formalism of attack trees, by including not only the actions of an attacker, but also possible counteractions of a defender. Hence, an ADTree can be seen as a game between two opposite players: an attacker trying to compromise a system and a defender trying to protect the system [16]. The root of the tree represents the main goal of one of the players. Using an AND-OR tree structure, this main goal is then refined into less complex subgoals that need to be reached in order to achieve the objective of the root. Every node of an ADTree can also be counteracted. This is depicted using a subtree rooted in a node of the opposite player. Since interactions between an attacker and a defender are modeled explicitly in ADTrees, the extended formalism allows for

* This is an extended version of the tool demonstration paper accepted for publication at QEST 2013 [15]. The research leading to the results presented in this work received funding from the Fonds National de la Recherche Luxembourg under the grants C08/IS/26 and PHD-09-167 and the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 318003 (TREsPASS).

a more thorough and accurate security analysis compared to attack trees, without however requiring additional computational power [20]. A toy example of an ADTree, representing how to attack a server, is given in Figure 2.

Theoretical foundations of the ADTree methodology, including a graphical and a term-based syntax as well as numerous formal semantics, have been introduced in [17]. A mathematical framework for quantitative evaluation of ADTrees has been developed in [18]. It is based on the notion of attributes, which allow us to formalize and specify relevant security metrics. Standard quantitative analysis of ADTrees relies on a step-wise computation procedure. Numerical values are assigned to all atomic actions, represented by the non-refined nodes[1]. The values for the remaining nodes, including the root of the tree, are deduced automatically in a bottom-up way. This bottom-up algorithm makes use of attribute domains which specify operators to be used while calculating values for different node configurations of an ADTree. In [19], we have developed guidelines for intuitive and formal specification and classification of the most popular measures for attack trees and ADTrees. There, we also provide guidelines for how to properly specify an attribute, so that it can be evaluated on ADTrees using the bottom-up algorithm.

1.2 Motivation

While validating the ADTree approach on a large industrial case study [4], it became apparent that the practical use of the methodology requires dedicated tool support. Lack of such support may result in numerous modeling difficulties and computational errors. On the one hand, there exist a number of commercial software applications for attack tree-like modeling, including SecurITree [3], and AttackTree+ [13]. However, these are closed source tools and their use is not free of charge. On the other hand, existing academic software, such as SeaMonster [26], does not support quantitative analysis and uniformly integrated defenses.

The above observations motivated the development of the ADTool, which

- is a free and open source application;
- supports quantitative and qualitative analysis of tree-based models integrating attack and defense components;
- is based on well-founded formal framework;
- guides the user in constructing syntactically correct models;
- allows to draw visually appealing trees;
- facilitates sharing, management and updating of the models;
- helps in verification of numerical values provided by the users;
- automates computation of security related parameters.

This paper provides an overview of the main features, practical capabilities, and the architecture of the ADTool. For a complete description of how to use the tool, we refer to the ADTool manual [22].

2 Main features of the ADTool

The goal of the ADTool is to provide security consultants as well as academic researchers with a rigorous but user-friendly application that supports security analysis

[1] Unlike in attack trees, a non-refined node of an ADTree is not necessarily a leaf. Such a node does not have any refining children but can still be countered.

and risk assessment process using attack–defense trees. It integrates two crucial modeling aspects: the creation of security models and their quantitative analysis. From a formal perspective, attack trees [31,25], protection trees [10], and defense trees [5] are instances of attack–defense trees. Thus, the ADTool can also be employed to automate and facilitate the usage of all aforementioned formalisms.

2.1 Security modeling using ADTool

One of the main features of the ADTool is its user-friendliness. The tool is guiding the user in constructing models that comply with the graphical ADTree language, as described in [18]. All options that allow to modify or refine a given component of a model can be accessed by right-clicking the node, as shown in Figure 1. Alternatively, intuitive keyboard shortcuts can be used to create, alter or remove a subtree [22].

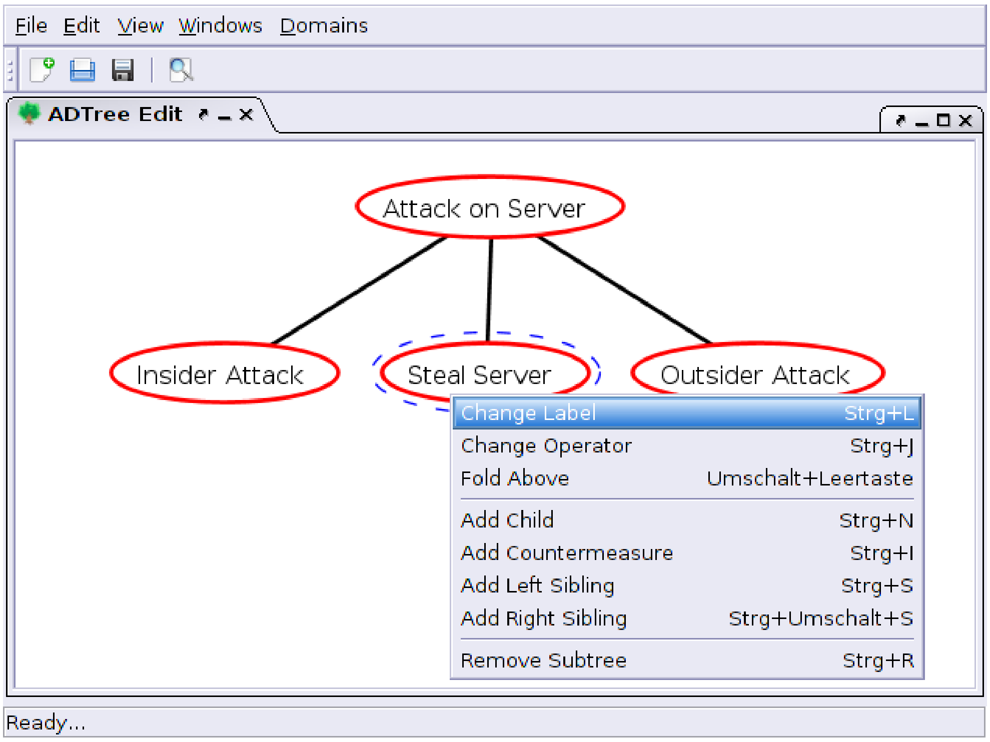

Fig. 1. Creating an ADTree in ADTool.

An improved version of Walker's algorithm [34], with enhancements suggested by Buchheim et al. [6], has been implemented in the ADTool to produce trees having an appealing layout. Furthermore, when an ADTree is built, the corresponding attack–defense term (ADTerm) is immediately displayed, as shown in Figure 2. ADTerms form a compact, algebraic representation of ADTrees. In order to link a term model with its graphical counterpart, the shortest tree edit distance algorithm [9,29] has been implemented. It ensures that when an ADTerm is modified, the corresponding ADTree is adapted accordingly.

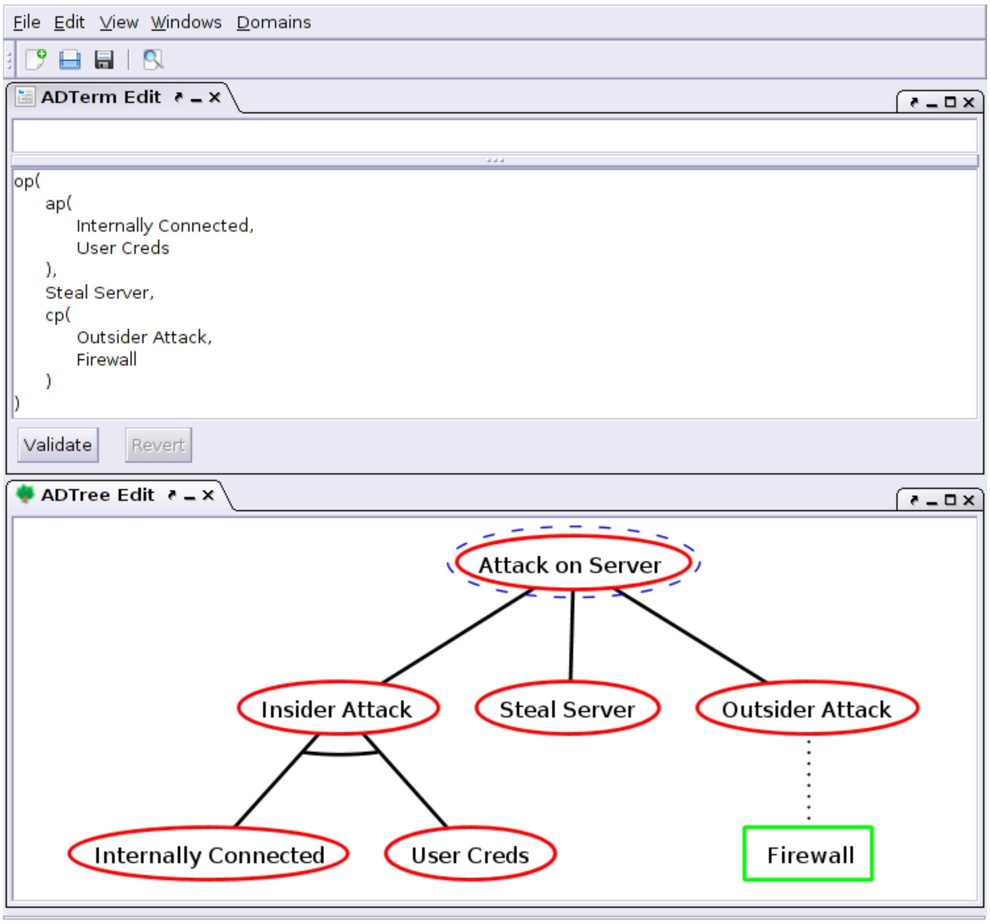

Fig. 2. An ADTree modeled in ADTool.

Finally, the ADTool provides advanced features for model manipulation and management. Folding, expanding and zooming options make the analysis of large models possible. Temporarily hiding parts of a tree permits users to focus on the displayed components. This is highly appreciated during industrial meetings and presentations. ADTrees created with the ADTool can be saved as special .adt files, which enables their reuse and modification. Models can also be exported to vector graphics files (pdf), raster graphics files (png, jpeg) and LaTeX files (tex). Resulting figures can be used as illustrations in scientific and industrial presentations, research papers and posters. A dedicated option, illustrated in Figure 3, makes it possible to print trees on a specified number of pages, which enhances readability of large-scale models.

2.2 Quantitative analysis using ADTool

The bottom-up algorithm for evaluation of attributes on ADTrees has been implemented in the ADTool. Supported measures include: attributes based on real values (e.g., time, cost, probability), attributes based on levels (e.g., required skill level, reachability of the goal in less than k units of time), and Boolean properties (e.g.,

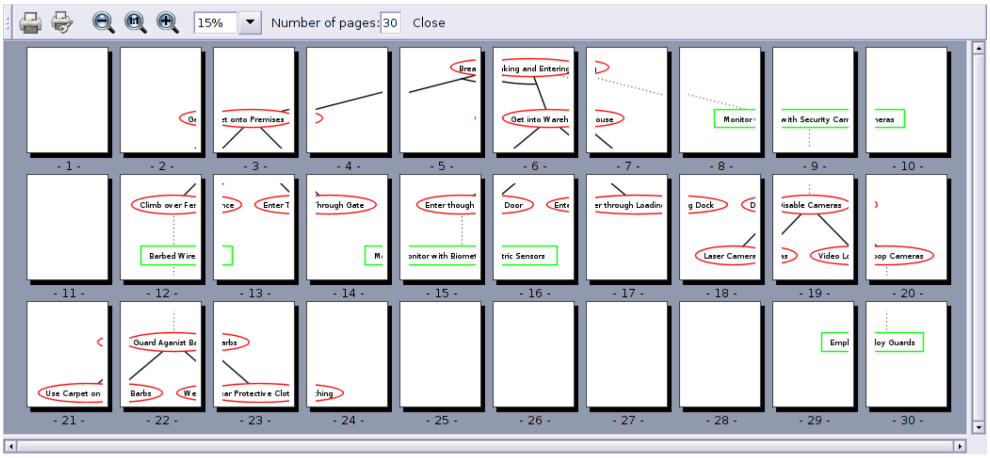

Fig. 3. Large-scale printing using ADTool.

satisfiability of a scenario). The implemented measures can be computed from the point of view of an attacker (e.g., the cost of an attack), of a defender (e.g., the cost of defending a system), or relate to both of them (e.g., overall maximum power consumption). By using different attribute domains, we distinguish between situations when actions are executed simultaneously and when they need to be performed sequentially. The choice of an attribute is illustrated in Figure 4.

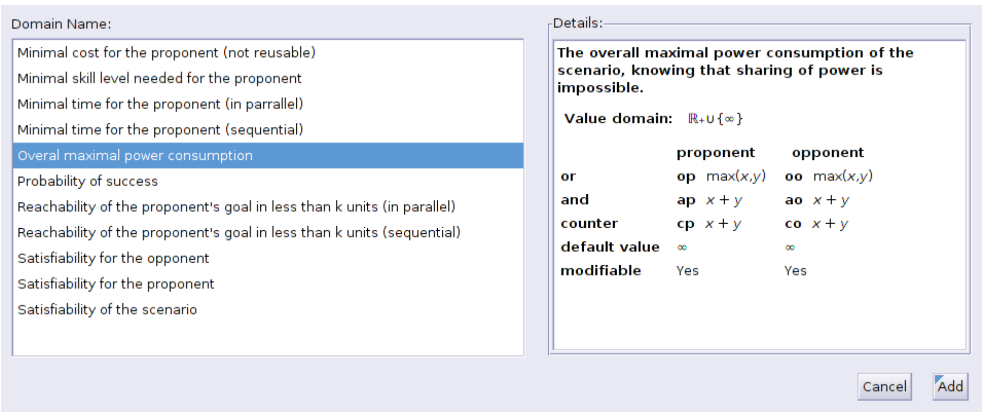

Fig. 4. Attribute selection in ADTool.

After a user has selected an attribute, the tool decorates the ADTree with default values representing the worst case scenario, e.g., infinite cost or maximal required skill level. The user then customizes the inputs for the relevant non-refined nodes directly on the tree or uses an overview table, as shown in Figure 5. The use of the table is particularly helpful in case of large models. The tool ensures that the provided values are consistent and belong to a specified value domain. This is especially important when several specialists supply values for different parts of the tree. Nodes labeled with the same name, i.e., representing the same action, automatically receive the same value. Such a design choice is consistent with the ADTree methodology, as specified in [18]. If a value for a non-refined node is modified, ADTool automatically computes the values of the remaining nodes using the bottom-up algorithm. By restricting the user input to the minimum and by automating computations, we avoid calculation errors.

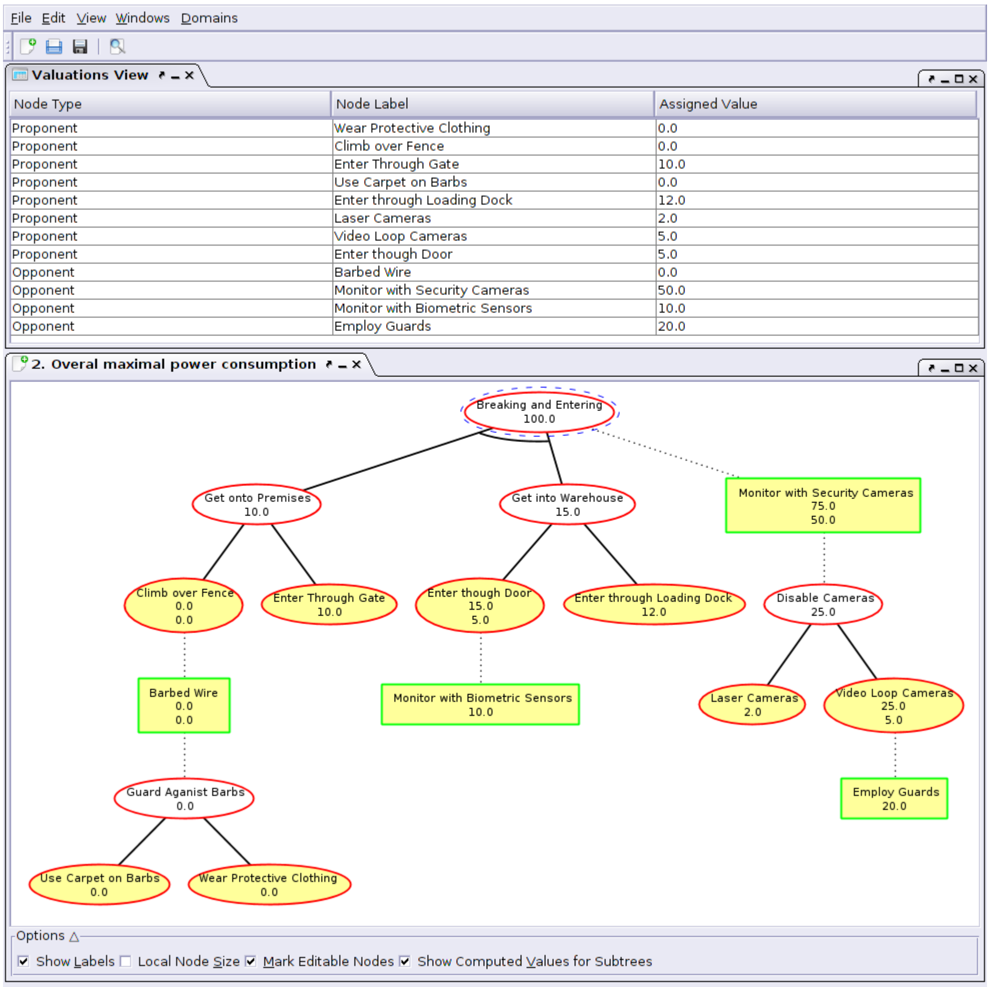

Fig. 5. Attribute evaluation in the ADTool.

2.3 Evaluation of ADTool

The ADTool easily handles trees containing a few thousand nodes. The complexity of the bottom-up algorithm for evaluation of attributes on ADTrees is linear with respect to the size of the tree, i.e., its number of nodes, and the computations are performed instantaneously. Thus, the quantitative analysis using ADTool scales very well. The limiting factor of the application is the graphical display of the models. For trees of more than ten thousand nodes, a delay of about five seconds occurs when a new node is added. This is due to the recalculation of the positions of some nodes. ADTool is programmed in Java. Implementation in C++ would run faster, but its development overhead would be much higher.

ADTool presents a number of advantages compared to existing tools employed for attack tree-based modeling. Graphical security models are usually prepared with the help of non-dedicated drawing tools. In this case, the user himself needs to make sure that the models he creates are syntactically correct. Since ADTool guides the user in creating only well-formed models, its employment does not require an in-depth knowledge of the technical details of the underlying methodology.

Another strong point of our application is that it supports both: creation and management of graphical models as well as their quantitative analysis. Existing prototype tools usually concentrate on one of the above aspects only. On the one hand, applications such as SeaMonster [26] support creation of security models, but they do not provide means to perform computations. On the other hand, most academic tools designed for quantitative analysis are command-line programs and they do not support visual representation of the analyzed models.

Finally, the ADTree methodology and the computation procedures implemented in ADTool have sound theoretical foundations [14]. In particular, the meaning of all attributes supported by the tool is clearly defined and the underlying mathematical models are well-studied from the formal point of view [17,18,19]. Together with the fact that ADTool is a free and open source application, this makes it a good alternative for commercial software for attack tree-based modeling, such as [3,13], etc.

3 Implementation Characteristics

The application has been written in a modular way with a clear distinction between the GUI and the Implementation Model. An overview of the ADTool architecture is depicted in Figure 6. The Implementation Model consists of the Tree Model (which

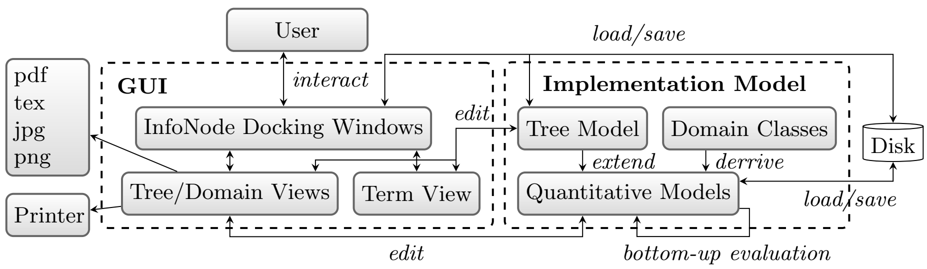

Fig. 6. An overview of the ADTool architecture

stores the basic tree structure), Domain Classes (defining the implemented attribute domains), and Quantitative Models (which are derived from Domain Classes and contain inserted and computed values). The functionality of the tool can easily be extended by defining new attributes. For this purpose, a new Domain Class needs to be created and compiled. Domain Classes have been designed to be simple, in order to make it possible for a user with minimal knowledge of Java to add a new domain. Due to the use of Java reflection, no recompilation or other modifications of the program are required after adding a new Domain Class.

The ADTool is open source software that runs on all common operating systems (Windows, Linux, Mac OS). It is implemented in Java and its source code is about 16 000 LOC. In order to run the program, JDK 6 or later is required. Additionally the ADTool depends on the following free libraries: abego TreeLayout [1], implementing an efficient and customizable tree layout algorithm in Java, and InfoNode Docking Windows [27], a pure Java Swing based docking windows framework, allowing to set up windows in a flexible way and to save and restore their layout. The ADTool is available for download at `http://satoss.uni.lu/software/adtool/`. It can also be launched as an online application that uses the Java Web Start technology [28].

4 Conclusion

The ADTool is free and open source software for the creation and quantitative analysis of tree-based security models integrating attack and defense components. The tool has been extensively tested and has proven to be able to handle realistic models of large size. Its modular architecture allows for an easy extension of the application's capabilities.

The ADTool is currently used in a case study where quantitative threat analysis of an electronic voting system is performed. Furthermore, we are working on combining the attack–defense tree methodology with Bayesian Networks, to make probabilistic reasoning about scenarios involving dependent actions possible. Related theoretical findings and newly identified features will be implemented in the next versions of the ADTool.

References

1. abego Software: abego TreeLayout. `http://code.google.com/p/treelayout/`, accessed October 5, 2012
2. Aijaz, A., Bochow, B., Dötzer, F., Festag, A., Gerlach, M., Kroh, R., Leinmüller, T.: Attacks on Inter Vehicle Communication Systems - an Analysis. In: 3rd International Workshop on Intelligent Transportation. pp. 189–194 (2006)
3. Amenaza: SecurITree. `http://www.amenaza.com/` (2001–2012), accessed May 5, 2013
4. Bagnato, A., Kordy, B., Meland, P.H., Schweitzer, P.: Attribute Decoration of Attack–Defense Trees. International Journal of Secure Software Engineering (IJSSE) 3(2), 1–35 (2012)
5. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense Trees for Economic Evaluation of Security Investments. In: ARES'06. pp. 416–423. IEEE Computer Society (2006)
6. Buchheim, C., Jünger, M., Leipert, S.: Drawing rooted trees in linear time. Software: Practice and Experience 36(6), 651–665 (2006), `http://dx.doi.org/10.1002/spe.v36:6`
7. Buldas, A., Mägi, T.: Practical Security Analysis of E-Voting Systems. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) International Workshop on Security (IWSEC'07). LNCS, vol. 4752, pp. 320–335. Springer (2007)
8. Byres, E.J., Franz, M., Miller, D.: The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems. In: International Infrastructure Survivability Workshop (IISW'04), Institute of Electrical and Electronics Engineers (2004)

9. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An Optimal Decomposition Algorithm for Tree Edit Distance. ACM Trans. Algorithms 6(1), 2:1–2:19 (2009)
10. Edge, K.S., Dalton II, G.C., Raines, R.A., Mills, R.F.: Using Attack and Protection Trees to Analyze Threats and Defenses to Homeland Security. In: MILCOM. pp. 1–7. IEEE (2006)
11. Eom, J.H., Park, M.W., Park, S.H., Chung, T.M.: A Framework of Defense System for Prevention of Insider's Malicious Behaviors. In: 13th International Conference on Advanced Communication Technology (ICACT'11). pp. 982–987 (2011)
12. Henniger, O., Apvrille, L., Fuchs, A., Roudier, Y., Ruddle, A., Weyl, B.: Security requirements for automotive on-board networks. In: 9th International Conference on Intelligent Transport Systems Telecommunications (ITST'09). pp. 641–646 (2009)
13. Isograph: AttackTree+. `http://www.isograph-software.com/2011/software/attacktree/`, accessed May 29, 2013
14. Kordy, B., Kordy, P., Mauw, S., Radomirović, S., Schweitzer, P., Weber, J.P.: The ATREES Project, funded by the Fonds National de la Recherche, Luxembourg under grants C08/IS/26 and PHD-09-167. `http://satoss.uni.lu/projects/atrees/` (2009–2012), accessed May 29, 2013
15. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: ADTool: Security Analysis with Attack–Defense Trees. In: Proceedings of the 10th International Conference on Quantitative Evaluation of SysTems (QEST'13). LNCS, Springer (2013)
16. Kordy, B., Mauw, S., Melissen, M., Schweitzer, P.: Attack–Defense Trees and Two-Player Binary Zero-Sum Extensive Form Games Are Equivalent. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec'10. LNCS, vol. 6442, pp. 245–256. Springer (2010)
17. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of Attack–Defense Trees. In: Degano, P., Etalle, S., Guttman, J.D. (eds.) FAST'10. LNCS, vol. 6561, pp. 80–95. Springer (2011)
18. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack–Defense Trees. Journal of Logic and Computation pp. 1–33 (2012), available online `http://logcom.oxfordjournals.org/content/early/2012/06/21/logcom.exs029.short?rss=1`
19. Kordy, B., Mauw, S., Schweitzer, P.: Quantitative Questions on Attack–Defense Trees. In: ICISC'12. LNCS, vol. 7839, pp. 49–64. Springer (2013)
20. Kordy, B., Pouly, M., Schweitzer, P.: Computational Aspects of Attack–Defense Trees. In: Security & Intelligent Information Systems. LNCS, vol. 7053, pp. 103–116. Springer (2011)
21. Kordy, P., Schweitzer, P.: The ADTool. `http://satoss.uni.lu/software/adtool` (2012), accessed May 29, 2013
22. Kordy, P., Schweitzer, P.: The ADTool Manual. `http://satoss.uni.lu/software/adtool/manual.pdf` (2012), accessed May 28, 2013
23. Lazarus, E.L., Dill, D.L., Epstein, J., Hall, J.L.: Applying a Reusable Election Threat Model at the County Level. In: Proceedings of the 2011 Conference on Electronic voting Technology/Workshop on Trustworthy Elections. pp. 1–14. EVT/WOTE'11, USENIX Association (2011)
24. Lin, X., Zavarsky, P., Ruhl, R., Lindskog, D.: Threat Modeling for CSRF Attacks. In: International Conference on Computational Science and Engineering (CSE'09). vol. 3, pp. 486 –491 (2009)
25. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D., Kim, S. (eds.) ICISC'05. LNCS, vol. 3935, pp. 186–198. Springer (2006), `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.1056`
26. Meland, P.H.: SeaMonster. `https://sourceforge.net/projects/seamonster/` (2010), accessed February 24, 2012
27. NNL Technology AB: InfoNode Docking Windows. `http://www.infonode.net/index.html?idw`, accessed December 5, 2012
28. Oracle: Java Web Start. `https://www.java.com/en/download/faq/java_webstart.xml`, accessed January 12, 2013

29. Pawlik, M., Augsten, N.: RTED: A Robust Algorithm for the Tree Edit Distance. Proceedings of the VLDB Endowment 5(4), 334–345 (2011)
30. Reddy, K., Venter, H.S., Olivier, M.S., Currie, I.: Towards Privacy Taxonomy-Based Attack Tree Analysis for the Protection of Consumer Information Privacy. In: Korba, L., Marsh, S., Safavi-Naini, R. (eds.) PST'08. pp. 56–64. IEEE (2008)
31. Schneier, B.: Attack Trees. Dr. Dobb's Journal of Software Tools 24(12), 21–29 (1999), `http://www.ddj.com/security/184414879`
32. Ten, C.W., Liu, C.C., Manimaran, G.: Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. In: Power Engineering Society General Meeting, IEEE. pp. 1–8 (2007)
33. Tidwell, T., Larson, R., Fitch, K., Hale, J.: Modeling Internet Attacks. In: Proceedings of the 2nd IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '01). pp. 54–59 (2001)
34. Walker II, J.Q.: A Node-positioning Algorithm for General Trees. Software: Practice and Experience 20(7), 685–705 (1990), `http://dx.doi.org/10.1002/spe.4380200705`